# LIBRARY MANAGEMENT SYSTEM USING LINKED LIST

*Submitted by*

**V.H.N.M.SWAMY(RA2311051010043)**

*Under the Guidance of*

**Dr. Rajkumar R.**
**Assistant Professor, Data Science and Business Systems**

*In partial satisfaction of the requirements for the degree of*

**BACHELORS OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE AND ENGINEERING**
**DATA SCIENCE**


**DATA SCIENCE AND BUSINESS SYSTEMS**
**COLLEGE OF ENGINEERING AND TECHNOLOGY SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**
**KATTANKULATHUR - 603203**

**October 2024**

SRM INSTITUTION OF SCIENCE AND TECHNOLOGY KATTANKULATHUR-603203

# BONAFIDE CERTIFICATE

Certified that this Course Project Report titled **"LIBRARY MANAGEMENT SYSTEM"** is the Bonafide work done by **Sai Preetham Jaini (RA2311056010046)** of II Year/ III Sem B.TECH – CSEDS who carried out under my supervision for the course **21CSC201J DATA STRUCTURES AND ALGORITHMS**. Certified further, that to the best of my knowledge the work reported herein does not form part of any other work.

**SIGNATURE**

Faculty In-Charge
**Dr. Rajkumar R HEAD OF THE DEPARTMENT**
Department of DSBS
SRM Institute of Science and Technology
Kattankulathur Campus, Chennai

AssistantProfessor **Dr. Kavitha V.**
Professor and Head ,
Department of DSBS,
SRM Institute of Science and Technology
Kattankulathur Campus, Chennai

# DATA STRUCTURES & ALGORITHMS



## 1.PROBLEM STATEMENT

**Problem Statement:** Library Management System that utilizes a linked list data structure to manage and keep track of book records. The system should allow efficient management of library operations such as adding new books, deleting books, searching for books, and updating book information.

**Objective:** The goal is to develop an efficient and simple way to manage book records using a linked list, providing basic operations for library staff and users to manage and access book information.

The linked list structure will allow dynamic addition and removal of book records, ensuring optimal memory usage compared to other data structures like arrays. Implement a linked list to store book information (e.g., title, author, ISBN, availability) efficiently. Design functions to insert new books and delete books from the list with minimal time complexity. Utilize the dynamic nature of linked lists to avoid memory wastage by allocating memory only when a new book is added. Implement a function to traverse the linked list and search for books based on various criteria.

## 2. DATA STRUCTURES USED

**Linked List Data Structure :** A linked list is a linear data structure used to store a collection of elements called nodes, where each node contains data and a reference (or pointer) to the next node in the sequence. Unlike arrays, linked lists do not store elements in contiguous memory locations, making them more flexible for operations involving dynamic memory allocation.

**Key Components of a Linked List:**

**Node:** Each element in the linked list is a node, which typically consists of Data and Pointer

**Head:** The first node in the linked list.

**Tail :** The last node which usually points to null.

**Basic Operations on a Linked List:**

1. Insertion
2. Deletion
3. Searching
4. Updating
5. Traversal

## 3. JUSTIFICATION

**1.Dynamic Memory Allocation**

- **Justification**: Linked lists offer dynamic memory allocation, which is ideal for a system where the number of books or records can change frequently (adding or removing items).
- **Elaboration**: In an LMS, the number of books isn't fixed. A linked list allows for flexible resizing without the overhead of reallocating or resizing as would be necessary in a contiguous data structure like arrays.

**2. Efficient Insertion and Deletion**

- **Justification**: Linked lists allow for efficient insertion and deletion operations, especially in scenarios where modifications to the data (e.g., adding a new book or removing a borrowed book) occur frequently.
- **Elaboration**: Unlike arrays where inserting or deleting an element requires shifting elements, linked lists can handle these operations in

**O(1)** time if pointers are already at the position of modification. This is useful for managing a dynamic collection of books or users.

**3.Modular Operations**

- **Justification**: Linked lists make it easy to create modular and extendable code, a valuable aspect for developing a library management system with different features (e.g., book reservation, due-date tracking).
- **Elaboration**: Linked lists can be used to implement queues or stacks for handling functions.

**4.Implementation of Various Operations**

- **Justification**: Linked lists support efficient traversal and can be easily tailored to create structures like doubly linked lists or circular linked lists for advanced features.
- **Elaboration**:
  - **Singly Linked List**: Useful for representing a basic collection of books with simple forward traversal.

# 4.TOOLS USED TO IMPLEMENT

**C Programming Language:**

C is the primary programming language used for writing the code. It is a versatile language that offers a rich set of features for system-level programming.

**C Compiler:**

You would need a C compiler to translate your C code into machine code that the computer can execute. Common C compilers include GCC (GNU Compiler Collection), Clang, and Microsoft Visual C.

**Integrated Development Environment (IDE):**

While you can write C code using a simple text editor and compile it through the command line, many developers prefer using an integrated development environment for a more streamlined development process. Some popular C **IDEs include:** Code block

# 5.SOURCE CODE

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct Book {
    int id;
    char title[100];
    char author[100];
    int isIssued;
    struct Book* next;
};
struct Book* head = NULL;
void addBook(int id, char* title, char* author) {
    struct Book* newBook = (struct Book*)malloc(sizeof(struct Book));
    newBook->id = id;
    strcpy(newBook->title, title);
    strcpy(newBook->author, author);
    newBook->isIssued = 0;
    newBook->next = NULL;

    if (!head) {
        head = newBook;
    } else {
        struct Book* temp = head;
        while (temp->next) {
            temp = temp->next;
        }
        temp->next = newBook;
    }
    printf("Book added successfully.\n");
}
struct Book* searchBook(int id) {
    struct Book* temp = head;
    while (temp) {
        if (temp->id == id) {
            return temp;
        }
        temp = temp->next;
```

```
    }
    return NULL;
}
struct Book* searchBookByTitle(char* title) {
    struct Book* temp = head;
    while (temp) {
        if (strcmp(temp->title, title) == 0) {
            return temp;
        }
        temp = temp->next;
    }
    return NULL;
}
void issueBook(int id, char* studentName) {
    struct Book* book = searchBook(id);
    if (book && !book->isIssued) {
        book->isIssued = 1;
        printf("Book issued to %s.\n", studentName);
    } else {
        printf("Book not available for issuing.\n");
    }
}
void returnBook(int id) {
    struct Book* book = searchBook(id);
    if (book && book->isIssued) {
        book->isIssued = 0;
        printf("Book returned successfully.\n");
    } else {
        printf("Book is not issued or doesn't exist.\n");
    }
}
void listBooks() {
    struct Book* books[100];
    int count = 0;
    struct Book* temp = head;

    while (temp) {
```

```
        books[count++] = temp;
        temp = temp->next;
    }
    for (int i = 0; i < count - 1; i++) {
        for (int j = 0; j < count - i - 1; j++) {
            if (strcmp(books[j]->title, books[j + 1]->title) > 0) {
                struct Book* temp = books[j];
                books[j] = books[j + 1];
                books[j + 1] = temp;
            }
        }
    }
    for (int i = 0; i < count; i++) {
        printf("ID: %d, Title: %s, Author: %s, Status: %s\n",
               books[i]->id, books[i]->title, books[i]->author,
               books[i]->isIssued ? "Issued" : "Available");
    }
}
void deleteBook(int id) {
    if (!head) return;

    if (head->id == id) {
        struct Book* toDelete = head;
        head = head->next;
        free(toDelete);
        printf("Book deleted successfully.\n");
        return;
    }
    struct Book* temp = head;
    while (temp->next && temp->next->id != id) {
        temp = temp->next;
    }
    if (temp->next) {
        struct Book* toDelete = temp->next;
        temp->next = temp->next->next;
        free(toDelete);
        printf("Book deleted successfully.\n");
```

```
    } else {
        printf("Book not found.\n");
    }
}
int main() {
    int choice, id;
    char title[100], author[100], studentName[100];

    while (1) {
        printf("\nLibrary Management System\n");
        printf("1. Add New Book\n");
        printf("2. Search Book by ID\n");
        printf("3. Search Book by Title\n");
        printf("4. Issue Book\n");
        printf("5. Return Book\n");
        printf("6. List All Books\n");
        printf("7. Delete Book\n");
        printf("8. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        getchar();

        switch (choice) {
            case 1:
                printf("Enter Book ID: ");
                scanf("%d", &id);
                getchar();
                printf("Enter Book Title: ");
                fgets(title, sizeof(title), stdin);
                title[strcspn(title, "\n")] = '\0';
                printf("Enter Book Author: ");
                fgets(author, sizeof(author), stdin);
                author[strcspn(author, "\n")] = '\0';
                addBook(id, title, author);
                break;
            case 2:
                printf("Enter Book ID: ");
```

```
            scanf("%d", &id);
            {
                struct Book* book = searchBook(id);
                if (book) {
                    printf("ID: %d, Title: %s, Author: %s, Status: %s\n",
                           book->id, book->title, book->author,
                           book->isIssued ? "Issued" : "Available");
                } else {
                    printf("Book not found.\n");
                }
            }
            break;
        case 3:
            printf("Enter Book Title: ");
            getchar();
            fgets(title, sizeof(title), stdin);
            title[strcspn(title, "\n")] = '\0';
            {
                struct Book* book = searchBookByTitle(title);
                if (book) {
                    printf("ID: %d, Title: %s, Author: %s, Status: %s\n",
                           book->id, book->title, book->author,
                           book->isIssued ? "Issued" : "Available");
                } else {
                    printf("Book not found.\n");
                }
            }
            break;
        case 4:
            printf("Enter Book ID: ");
            scanf("%d", &id);
            getchar();
            printf("Enter Student Name: ");
            fgets(studentName, sizeof(studentName), stdin);
            studentName[strcspn(studentName, "\n")] = '\0';
            issueBook(id, studentName);
            break;
```

```
            case 5:
                printf("Enter Book ID: ");
                scanf("%d", &id);
                returnBook(id);
                break;
            case 6:
                listBooks();
                break;
            case 7:
                printf("Enter Book ID: ");
                scanf("%d", &id);
                deleteBook(id);
                break;
            case 8:
                printf("Exiting...\n");
                return 0;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }
    return 0;
}
```

# 5.CODE OUTPUT

```
Library Management System
1. Add New Book
2. Search Book by ID
3. Search Book by Title
4. Issue Book
5. Return Book
6. List All Books
7. Delete Book
8. Exit
Enter your choice: 1
Enter Book ID: 1234
Enter Book Title: Beautiful Day
Enter Book Author: Alex
Book added successfully.
```

```
Library Management System
1. Add New Book
2. Search Book by ID
3. Search Book by Title
4. Issue Book
5. Return Book
6. List All Books
7. Delete Book
8. Exit
Enter your choice: 2
Enter Book ID: 1234
ID: 1234, Title: Beautiful Day, Author: Alex, Status: Available
```

```
Library Management System
1. Add New Book
2. Search Book by ID
3. Search Book by Title
4. Issue Book
5. Return Book
6. List All Books
7. Delete Book
8. Exit
Enter your choice: 4
Enter Book ID: 1234
Enter Student Name: Preetham
Book issued to Preetham.
```

```
Library Management System
1. Add New Book
2. Search Book by ID
3. Search Book by Title
4. Issue Book
5. Return Book
6. List All Books
7. Delete Book
8. Exit
Enter your choice: 5
Enter Book ID: 1234
Book returned successfully.
```

```
Library Management System
1. Add New Book
2. Search Book by ID
3. Search Book by Title
4. Issue Book
5. Return Book
6. List All Books
7. Delete Book
8. Exit
Enter your choice: 6
ID: 1234, Title: Beautiful Day, Author: Alex, Status: Available
```

```
Library Management System
1. Add New Book
2. Search Book by ID
3. Search Book by Title
4. Issue Book
5. Return Book
6. List All Books
7. Delete Book
8. Exit
Enter your choice: 7
Enter Book ID: 1234
Book deleted successfully.
```

```
Library Management System
1. Add New Book
2. Search Book by ID
3. Search Book by Title
4. Issue Book
5. Return Book
6. List All Books
7. Delete Book
8. Exit
Enter your choice: 8
Exiting...
```